LIBRARY OF CONGRESS

Office of Business Enterprises
Duplication Services Section

THIS IS TO CERTIFY that the collections of the Library of Congress contain a bound volume entitled **REAL-TIME VIDEO COMPRESSION Techniques and Algorithms,** call number QA76.575.W47 1997, Copy 2, The attached – Cover Page, Title Page, Copyright Page, Table of Contents Pages, Preface page, Chapters 1,2,3 and Chapter 7 - are a true and complete representation from that work.

THIS IS TO CERTIFY FURTHER, that work is marked with a Library of Congress Copyright Office stamp dated November 25, 1996.

IN WITNESS WHEREOF, the seal of the Library of Congress is affixed hereto on May 18, 2018.

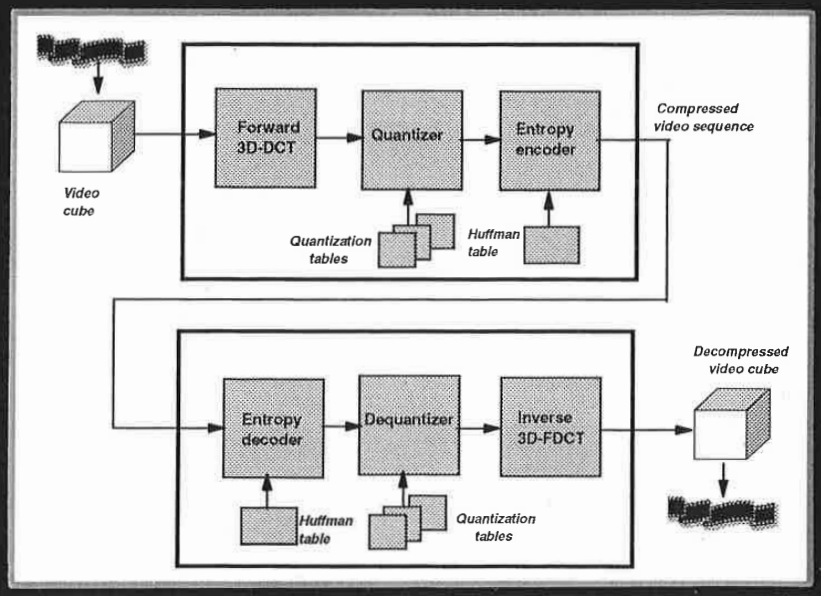
Deirdre Scott
Business Enterprises Officer
Office of Business Enterprises
Library of Congress

101 Independence Avenue, SE Washington, DC 20540–4917 Tel 202.707.5650 www.loc.gov; duplicationservices@loc.gov

REAL-TIME VIDEO COMPRESSION

Techniques and Algorithms

Raymond Westwater
Borko Furht



REAL-TIME VIDEO COMPRESSION
Techniques and Algorithms

by

Raymond Westwater
Borko Furht
Florida Atlantic University

KLUWER ACADEMIC PUBLISHERS
Boston / Dordrecht / London

Distributors for North America:
Kluwer Academic Publishers
101 Philip Drive
Assinippi Park
Norwell, Massachusetts 02061 USA

Distributors for all other countries:
Kluwer Academic Publishers Group
Distribution Centre
Post Office Box 322
3300 AH Dordrecht, THE NETHERLANDS

Library of Congress Cataloging-in-Publication Data

A C.I.P. Catalogue record for this book is available
from the Library of Congress.

Printed on acid-free paper.

Printed in the United States of America

CONTENTS

PREFACE

This book is on real-time video compression. Specifically, the book introduces the XYZ video compression technique, that operates in three dimensions, eliminating the overhead of motion estimation. First, video compression standards, MPEG and H.261/H.263, are described. They both use asymmetric compression algorithms, based on motion estimation. Their encoders are much more complex than decoders. The XYZ technique uses a symmetric algorithm, based on the Three-Dimensional Discrete Cosine Transform (3D-DCT). 3D-DCT was originally suggested for compression about twenty years ago, however at that time the computational complexity of the algorithm was to high, it required large buffer memory, and was not as effective as motion estimation. We have resurrected the 3D-DCT based video compression algorithm by developing several enhancements to the original algorithm. These enhancements made the algorithm feasible for real-time video compression in applications such as video-on-demand, interactive multimedia, and videoconferencing. The demonstrated results, presented in the book, suggest that the XYZ video compression technique is not only a fast algorithm, but also provides superior compression ratios and high quality of the video compared to existing standard techniques, such as MPEG and H.261/H.263. The elegance of the XYZ technique is in its simplicity, which leads to inexpensive VLSI implementation of a XYZ codec.

We would like to thank Jim Prince for conducting experiments in developing visually weighted quantizers for the XYZ algorithm, as well as a number of students from Florida Atlantic University, who participated in these experiments. We also want to thank Drs. Roy Levow, K. Genesan, and Matthew Evett, professors from Florida Atlantic University, Dr. Steve Rosenbaum from Cylex Systems, and Joshua Greenberg for constructive discussions during this project.

Raymond Westwater and Borko Furht
Boca Raton, July 1996.

1

THE PROBLEM OF VIDEO COMPRESSION

The problem of real-time video compression is a difficult and important one, and has inspired a great deal of research activity. This body of knowledge has been, to a substantial degree, embodied into the MPEG and H.261/H263 motion video standards. However, some important questions remain unexplored. This book describes one possible alternative to these standards that has superior compression characteristics while requiring far less computational power for its full implementation.

Since about 1989, moving digital video images have been integrated with programs. The difficulty in implementing moving digital video is the tremendous bandwidth required for the encoding of video data. For example, a quarter screen image (320 x 240 pixels) playing on an RGB video screen at full speed of 30 frames per second (fps) requires storage and transmission of 6.9 million bytes per second. This data rate is simply prohibitive, and so means of compressing digital video suitable for real-time playback are a necessary step for the widespread introduction of digital motion video applications.

Many digital video compression algorithms have been developed and implemented. The compression ratios of these algorithms varies according to the subjective acceptable level of error, the definition of the word compression, and who is making the claim. Table 1.1 summarizes video compression algorithms, their typical compression ratios reported in the literature, and their characteristics.

Compression Algorithm	Typical Compression Ratio	Characteristics
Intel RTV/Indeo	**3:1**	A 128X240 data stream is interpolated to 256X240. Color is subsampled 4:1. A simple 16 bit codebook is used without error correction. Frame differencing is used.
Intel PLV	**12:1**	A native 256X240 stream is encoded using vector quantization and motion compensation. Compression requires specialized equipment.
IBM Photomotion	**3:1**	An optimal 8-bit color palette is determined, and run-length encoding and frame differencing are used.
Motion JPEG	**10:1**	Uses 2-D DCT to encode individual frames. Gives good real-time results with inexpensive but special-purpose equipment. This technique supports random-access since no frame differencing is used.
Fractals	**10:1**	Fractals compress natural scenes well, but require tremendous computing power.
Wavelets	**20:1**	2-D and 3-D wavelets have been used in the compression of motion video. Wavelet compression is low enough in complexity to compress entire images, and therefore does not suffer from the boundary artifacts seen in DCT-based techniques.
H.261/H263	**50:1**	Real-time compression and decompression algorithm for video telecommunications. It is based on 2-D DCT with simple motion estimation between frames.
MPEG	**30:1**	Uses 2-D DCT with motion estimation and interpolation between frames. The MPEG standard is difficult and expensive to compress, but plays back in real-time with inexpensive equipment.

Table 1.1 Overview of video compression algorithms.

An ideal video compression technique should have the following characteristics:

- Will produce levels of compression rivaling MPEG without objectionable artifacts.
- Can be played back in real time with inexpensive hardware support.

- Can degrade easily under network overload or on a slow platform.
- Can be compressed in real time with inexpensive hardware support.

1.1 OVERVIEW OF VIDEO COMPRESSION TECHNIQUES

The JPEG still picture compression standard has been extremely successful, having been implemented on virtually all platforms. This standard is fairly simple to implement, is not computationally complex, and gets 10:1 to 15:1 compression ratios without significant visual artifacts. This standard is based upon entropy encoding of quantized coefficients of the discrete cosine transformation of 8x8 blocks of pixel data.

Figure 1.1 shows the block diagram of both the JPEG compression and decompression algorithms. A single frame is subdivided into 8x8 blocks, each of which is independently processed. Each block is transformed into DCT space, resulting in an 8x8 block of DCT coefficients. These coefficients are then quantized by integer division by constants. The quantizing constant for each DCT coefficient is chosen to produce minimal visual artifacts, while maximally reducing the representational entropy of the coefficients. The quantized coefficients are then entropy coded into a compressed data stream. The reduced entropy of the quantized coefficients is reflected in the higher compression ratio of the data.

The Motion JPEG (M-JPEG) uses the JPEG compression for each frame. It provides random access to individual frames, however the compression ratios are too low (same as in JPEG), because the technique does not take advantage of the similarities between adjacent frames.

The MPEG moving compression standard is an attempt to extend DCT-based compression into moving pictures. MPEG encodes frames by estimating the motion difference between the frames, and encoding the differences into roughly JPEG format. Unfortunately, motion estimation is computationally complex, requires specialized equipment to encode, and adds considerable complexity to the algorithm. Figure 1.2 illustrates the MPEG compression algorithm for predictive frames.

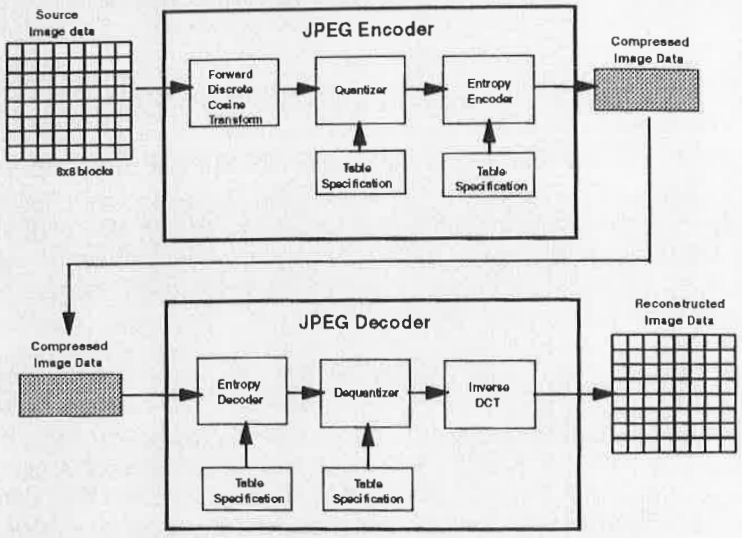


Figure 1.1 JPEG compression and decompression algorithms.

One of the most promising new technologies is wavelet-based compression [VK95]. Figure 1.3 illustrates a simple wavelet transform: subband decomposition. The image as a whole is subdivided into frequency subbands, which are then individually quantized. One of the most attractive features of this system is that it is applied to the image as a whole, thereby avoiding the edge artifacts associated with the block-based DCT compression schemes.

The wavelet transform can be applied to the time dimension as well. Experience has shown that this decomposition does not give as good compression results as motion compensation. As there are no other compression algorithms capable of such high compression ratios, MPEG is considered the existing "state-of-the-art".

The XYZ algorithm is a natural extension of the research that has been done in video compression. Much work has been done in the development of transform-based motion video compression algorithms, and in the development of quantizing factors based on the sensitivity of the human eye to various artifacts of compression.

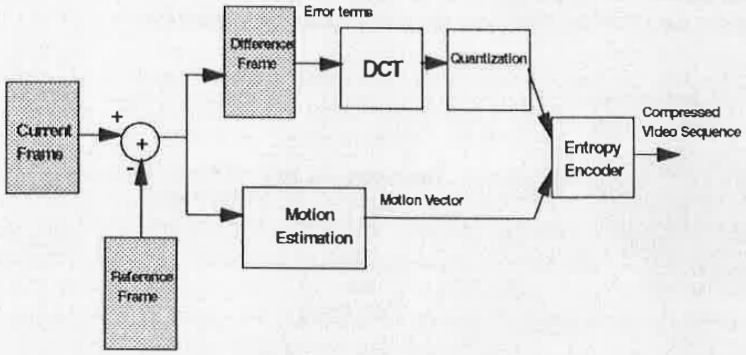


Figure 1.2 MPEG compression algorithm for predictive frames.
MPEG adds motion estimation to the JPEG model.

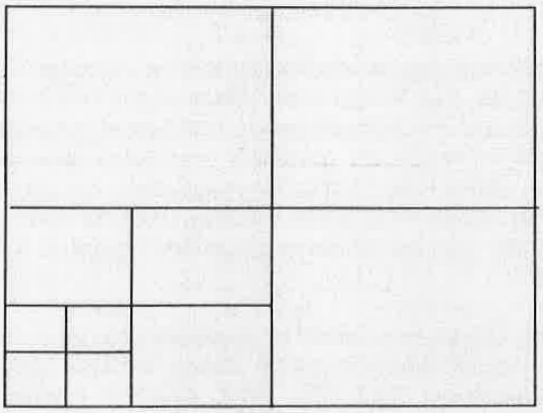


Figure 1.3 Octave-band or wavelet decomposition of a still image into unequal subbands.

XYZ compression is an alternative extension of DCT encoding to moving pictures. Sequences of eight frames are collected into a three-dimensional block to which a three-dimensional DCT will be applied. The transformed data is then quantized.

These quantizing constants are demonstrated to cause artifacts which are minimally visible. The resulting data stream is then entropy coded. This process strongly resembles the JPEG encoding process, as illustrated in Figure 1.4.

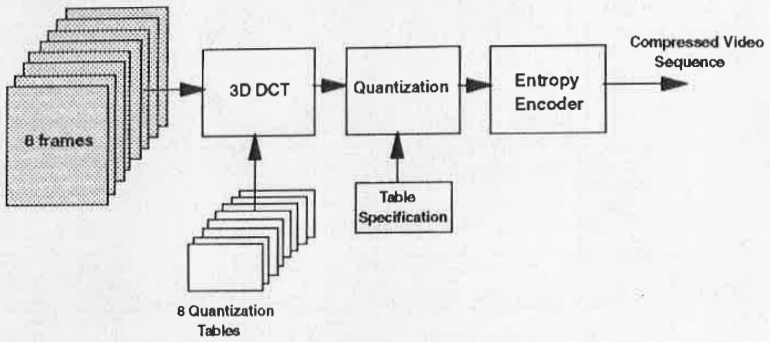


Figure 1.4 XYZ compression algorithm.

This algorithm is built upon a considerable body of published work. The three-dimensional DCT has been used to encode errors after motion estimation has been performed [RP77], and true three-dimensional DCT-based compression algorithms have been developed where the quantizers were based upon minimization of introduced mean square error [NA77]. These algorithms have fallen into disfavor because they were considered to require excessive computation, required too much buffer memory, and were not as effective as motion estimation. This book refutes these arguments.

Work in visibility of artifacts produced by quantization has also been done [CR90]. Visibility of two-dimensional quantization artifacts has been thoroughly explored for the DCT transforms space. The XYZ algorithm extends this work to quantization of three-dimensional DCT coefficients.

1.2 APPLICATIONS OF COMPRESSED VIDEO

Video compression techniques made feasible a number of applications. Four distinct applications of the compressed video can be summarized as: (a) consumer broadcast television, (b) consumer playback, (c) desktop video, and (d) videoconferencing.

Consumer broadcast television, which includes digital video delivery to homes, typically requires a small number of high-quality compressors and a large number of low-cost decompressors. Expected compression ratio is about 50:1.

Consumer playback applications, such as CD-ROM libraries and interactive games, also require a small number of compressors and a large number of low-cost decompressors. The required compression ratio is about 100:1.

Desktop video, which includes systems for authoring and editing video presentations, is a symmetrical application requiring the same number of encoders and decoders. The expected compression ratio is in the range from 5:1 to 50:1.

Videoconferencing applications also require the same number of encoders and decoders, and the expected compression ratio is about 100:1.

Application	Bandwidth	Standard	Size	Frame Rate [frames/sec]
Analog Videophone	5-10 Kbps	none	170x128	2-5
Low Bitrate Video Conferencing	26-64 Kbps	H.263	128x96 176x144	15-30
Basic Video Telephony	64-128 Kbps	H.261	176x144 352x288	10-20
Video Conferencing	>= 384 Kbps	H.261	352x288	15-30
Interactive Multimedia	1-2 Mbps	MPEG-1	352x240	15-30
Digital TV - NTSC	3-10 Mbps	MPEG-2	720x480	30
High Definition Television	15-80 Mbps	MPEG-2	1200x800	30-60

Table 1.2 Applications of the compressed video and current video compression standards.

Table 1.2 summarizes applications of the compressed video, by specifying current standards used in various applications, the required bandwidth, and typical frame sizes and frame rates.

1.3 IMAGE AND VIDEO FORMATS

A digital image represents a two-dimensional array of samples, where each sample is called a pixel. Precision determines how many levels of intensity can be represented, and is expressed as the number of bits/sample. According to precision, images can be classified into: (a) binary images, represented by 1 bit/sample, (b) computer graphics, represented by 4 bits/sample, (c) grayscale images, represented by 8 bits/sample, and color images, represented with 16, 24 or more bits/sample.

According to the trichromatic theory, the sensation of color is produced by selectively exciting three classes of receptors in the eye. In a RGB color representation system, shown in Figure 1.5, a color is produced by adding three primary colors: red, green, and blue (RGB). The straight line, where R=G=B, specifies the gray values ranging from black to white.

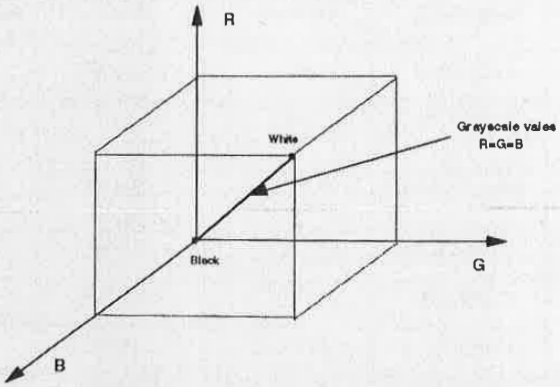


Figure 1.5 The RGB representation of color images.

Another representation of color images, YUV representation, describes luminance and chrominance components of an image. The luminance component provides a grayscale version of the image, while two chrominance components give additional information that converts the grayscale image to a color image. The YUV representation is more natural for image and video compression. The exact transformation from RGB to YUV representation, specified by the CCIR 601 standard, is given by the following equations:

$$Y = 0.299R + 0.587G + 0.114B \quad (1.1)$$

$$U = 0.564(B - Y) \quad (1.2)$$

$$V = 0.713(B - Y) \quad (1.3)$$

where Y is the luminance component, and U and V are two chrominance components.

An approximate RGB to YUV transformation is given as:

$$Y = 0.3R + 0.6G + 0.1B \quad (1.4)$$

$$U = B - Y \quad (1.5)$$

$$V = R - Y \quad (1.6)$$

This transformation has a nice feature that, when R+G+B, then Y=R=G=B, and U=V=0. In this case, the image is a grayscale image.

Color conversion from RGB to YUV requires several multiplications, which can be computationally expensive. An approximation, proposed in [W+94], can be calculated by performing bit shifts and adds instead multiplication operations. This approximation is defines as:

$$Y = \frac{R}{4} + \frac{G}{2} + \frac{B}{2} \quad (1.7)$$

$$U = \frac{B - Y}{2} \quad (1.8)$$

$$V = \frac{R - Y}{2} \quad (1.9)$$

This approximation also gives a simplified YUV to RGB transformation, expressed by:

$$R = Y + 2V \tag{1.10}$$

$$G = Y - (U + V) \tag{1.11}$$

$$B = Y + 2U \tag{1.12}$$

Another color format, referred to as YCbCr format, is intensively used for image compression. In YCbCr format, Y is the same as in a YUV system, however U and V components are scaled and zero-shifted to produce Cb and Cr, respectively, as follows:

$$Cb = \frac{U}{2} + 0.5 \tag{1.13}$$

$$Cr = \frac{V}{1.6} + 0.5 \tag{1.14}$$

In this way, chrominance components Cb and Cr are always in the range [0,1].

Computer Video Formats

Resolutions of an image system refers to its capability to reproduce fine detail. Higher resolution requires more complex imaging systems to represent these images in real time. In computer systems, resolution is characterized with number of pixels. Table 1.3 summarizes popular computer video formats, and related storage requirements.

Television Formats

In television systems, resolution refers to the number of line pairs resolved on the face of the display screen, expressed in cycles per picture height, or cycles per picture width. For example, the NTSC broadcast system in North America and Japan, denoted as 525/59.94, has about 483 picture lines.

The HDTV system will approximately double the number of lines of current broadcast television at approximately the same field rate. For example, a 1050x960 HDTV system will have 960 total lines. Spatial and temporal characteristics of conventional television systems (such as NTSC, SECAM, and PAL), and high-

definition TV systems (HDTV) are presented in Tables 1.4 and 1.5, respectively [BF91].

Computer Video Format	Resolution (pixels)	Colors (bits)	Storage Capacity Per Image
CGA - Color Graphics Adapter	320x200	4 (2 bits)	128,000 bits = 16 KB
EGA - Enhanced Graphics Adapter	640x350	16 (4 bits)	896,000 bits= 112 KB
VGA - Video Graphics Adapter	640x480	256 (8 bits)	2,457,600 bits= 307.2 KB
88514/A Display Adapter Mode	1024x768	256 (8 bits)	6,291,456 bits= 786.432 KB
XGA - Extended Graphics Array (a)	640x480	65,000 (24 bits)	6,291,456 bits= 786.432 KB
XGA - Extended Graphics Array (b)	1024x768	256 (8 bits)	6,291,456 bits= 786.432 KB
SVGA - Super VGA	1024x768	65,000 (24 bits)	2.36 MB

Table 1.3 Characteristics of various computer video formats.

System	Total Lines	Active Lines	Vertical Resolution	Optimal Viewing Distance [m]	Aspect Ratio	Horizontal Resolution	Total Picture Elements
HDTV USA	1050	960	675	2.5	16/9	600	720,000
HDTV Europe	1250	1000	700	2.4	16/9	700	870,000
NTSC	525	484	242	7.0	4/3	330	106,000
PAL	625	575	290	6.0	4/3	425	165,000
SECAM	625	575	290	6.0	4/3	465	180,000

Table 1.4 Spatial characteristics of television systems [BF91].

System	Total Channel Width [MHz]	Video Baseband Y [MHz]	Video Baseband R-Y [MHz]	Video Baseband B-Y [MHz]	Scanning Rate Camera [Hz]	Scanning Rate HDTV Display [Hz]	Scanning Rate Convent. Display [Hz]
HDTV USA	9.0	10.0	5.0	5.0	59.94	59.94	59.94
HDTV Europe	12.0	14.0	7.0	7.0	50	100	50
NTSC	6.0	4.2	1.0	0.6	59.94	NA	59.94
PAL	8.0	5.5	1.8	1.8	50	NA	50
SECAM	8.0	6.0	2.0	2.0	50	NA	50

Table 1.5 Temporal characteristics of television systems [BF91].

1.4 OVERVIEW OF THE BOOK

This book is divided into ten chapters:

1. **Video compression.** This current chapter introduces the problem of compressing motion video, illustrates the motivation for the 3-D solution chosen in the book, and briefly describes the proposed solution. Image and video formats are introduced as well.
2. **MPEG.** This chapter describes the MPEG compression standard. Important contributions in the field and related work are emphasized.
3. **H.261/H.263.** This chapter describes the compression standard for video telecommunications.
4. **XYZ compression.** The XYZ video compression algorithm is described in detail in this chapter. Both encoder and decoder are presented, as well as an example of compressing 8x8x8 video block.
5. **3-D DCT.** The theory of the Discrete Cosine Transform is developed and extended to three dimensions. A fast 3-D algorithm is developed.
6. **Quantization.** Discussion is presented on the issues of determining optimal quantizers using various error criteria. A model of Human Visual System is used to develop factors that weigh the DCT coefficients according to their relative visibility.
7. **Entropy coding.** A method for encoding the quantized coefficients is developed based on the stochastic behavior of the pixel data.

8. **VLSI architectures for XYZ codec.** Issues concerning real-time implementation of the XYZ compression algorithm are analyzed including the complexity of the algorithm and mapping the algorithm into various VLSI architectures.
9. **Results.** Obtained results of an implementation of the XYZ compression algorithm are presented.
10. **Conclusion.** Summary of contributions are outlined, emphasizing the real-time features of the compression algorithm, visual quality, and compression ratio. Directions for future research are given as well.

2

THE MPEG VIDEO COMPRESSION STANDARD

The Motion Picture Experts' Group was assembled by the International Standards Organization (ISO) to establish standards for the compression, encoding, and decompression of motion video. MPEG-1 [IS92b] is a standard supporting compression of image resolutions of approximately 352x288 at 30 fps into a data stream of 1.5 Mbps. This data rate is suitable for pressing onto CD-ROM. The MPEG-2 standard [IS93b] supports compression of broadcast television (704x576 at 30 fps) and HDTV (1920x1152 at 60 fps) of up to 60 Mpixels/sec (appx. 700 Mb) at compression ratios of roughly three times those expected of moving JPEG [IS92a] (playback rates of up to 80 Mbps).

The MPEG standard specifies the functional organization of a decoder. The data stream is cached in a buffer to reduce the effect of jitter in delivery and decode, and is demultiplexed into a video stream, an audio stream, and additional user-defined streams. The video stream is decoded into a "video sequence" composed of the sequence header and groups of pictures.

2.1 MPEG ENCODER AND DECODER

The specification of the MPEG encoder defines many compression options. While all of these options must be supported by the decoder, the selection of which options to support in compression is left to the discretion of the implementer. An MPEG encoder may choose compression options balancing the need for high compression

ratios against the complexity of motion compensation or adaptive quantization calculations. Decisions will be affected by such factors as:

- A need for real-time compression. MPEG algorithms are complex, and there may not be sufficient time to implement exotic options on a particular platform.
- A need for high compression ratios. For highest possible compression ratios at highest possible quality, every available option must be exercised.
- A need for insensitivity to transmission error. MPEG-2 supports recovery from transmission errors. Some error recovery mechanisms are implemented by the encoder.
- Fast algorithms. Development of fast algorithms may make compression options available that would otherwise be impractical.
- Availability of specialized hardware. Dedicated hardware may increase the performance of the encoder to the point that additional compression options can be considered.

In the MPEG standard, frames in a sequence are coded using three different algorithms, as illustrated in Figure 2.1.

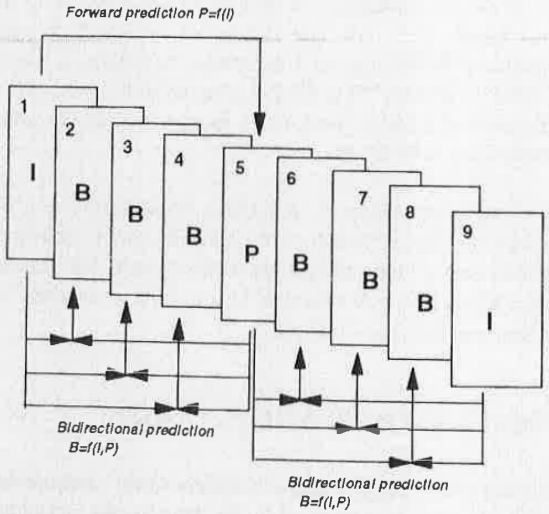


Figure 2.1 Types of frames in the MPEG standard.

I frames (intra frames) are self-contained and coded using a DCT-based technique similar to JPEG. I frames are used as random access points in MPEG streams, and they give the lowest compression ratios within MPEG.

P frames (predicted frames) are coded using forward predictive coding, where the actual frame is coded with reference to a pervious frame (I or P). This process is similar to H.261/H.263 predictive coding, except the previous frame is not always the closest previous frames, as in H.261/H.263 coding. The compression ratio of P frames is significantly higher than of I frames.

B frames (bidirectional or interpolated frames) are coded using two reference frames, a past and a future frame (which can be I or P frames). Bidirectional, or interpolated coding provides the highest amount of compression [Fur95b].

I, P, and B frames are described in more detail in Section 8.2. Note that in Figure 2.1, the first three B frames (2,3, and 4) are bidirectionally coded using the past frame I (frame 1), and the future frame P (frame 5). Therefore, the decoding order will differ from the encoding order. The P frame 5 must be decoded before B frames 2,3, and 4, and I frame 9 before B frames 6,7, and 8. If the MPEG sequence is transmitted over the network, the actual transmission order should be {1,5,2,,3,4,8,6,7,8}.

The MPEG application determines a sequence of I, P, and B frames. If there is a need for fast random access, the best resolution would be achieved by coding the whole sequence as I frames (MPEG becomes identical to Motion JPEG). However, the highest compression ratio can be achieved by incorporating a large number of B frames.

The block diagram of the MPEG encoder is given in Figure 2.2, while the MPEG decoder is shown in Figure 2.3.

I frames are created similarly to JPEG encoded pictures, while P and B frames are encoded in terms of previous and future frames. The motion vector is estimated, and the difference between the predicted and actual blocks (error terms) are calculated. The error terms are then DCT encoded and the entropy encoder is used to produce the compact code.

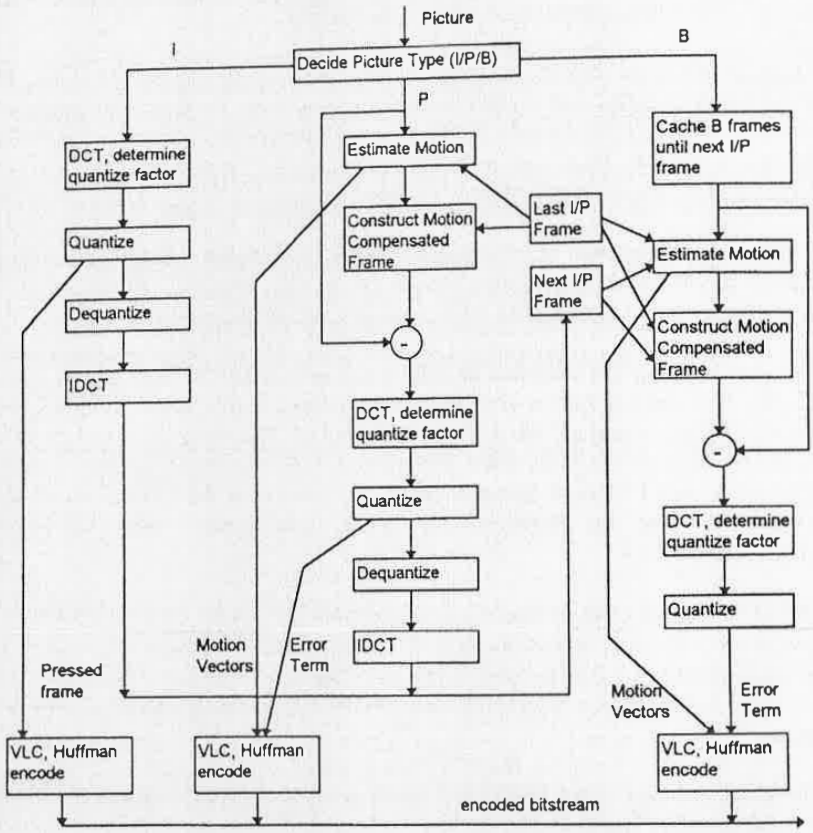


Figure 2.2 The block diagram of the MPEG encoder.

2.2 MPEG DATA STREAM

The MPEG specification defines a "video sequence" composed of a video sequence header and many Group-Of-Pictures (GOP), as illustrated in Figure 2.4. The video sequence header defines the video format, picture dimensions, aspect ratio, frame rate, and delivered data rate. Supported video formats include CCIR601, HDTV(16:9), and VGA. Supported chroma formats include "4:2:0" (YUV) and

"4:4:4" (RGB). A suggested buffer size for the video sequence is also specified, a number intended to buffer jitter caused by differences in decode time.

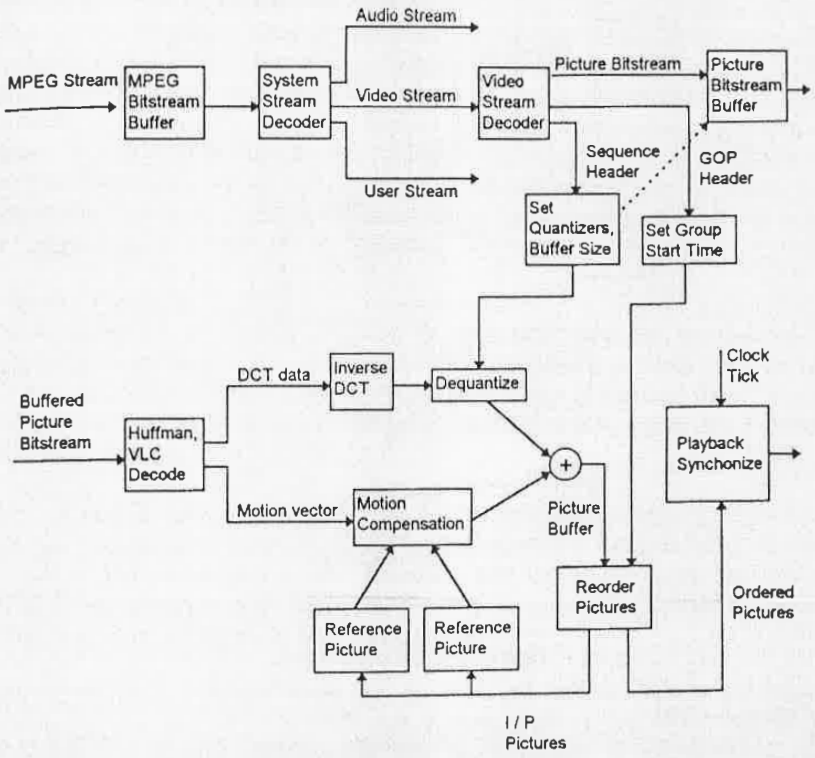


Figure 2.3 The block diagram of the MPEG decoder.

A GOP contains pictures that may be encoded into one of three supported compression formats. The GOP header contains a starting time for the group, and can therefore be used as a point of random access. Each frame within the GOP is numbered, and its number coupled with the GOP start time and the playback frame rate determines its playback time. Each picture is subdivided into "slices" and then into "macroblocks". A macroblock is composed of four 8x8 blocks of luminance data, and typically two 8x8 blocks of chrominance data, one Cr and one Cb.

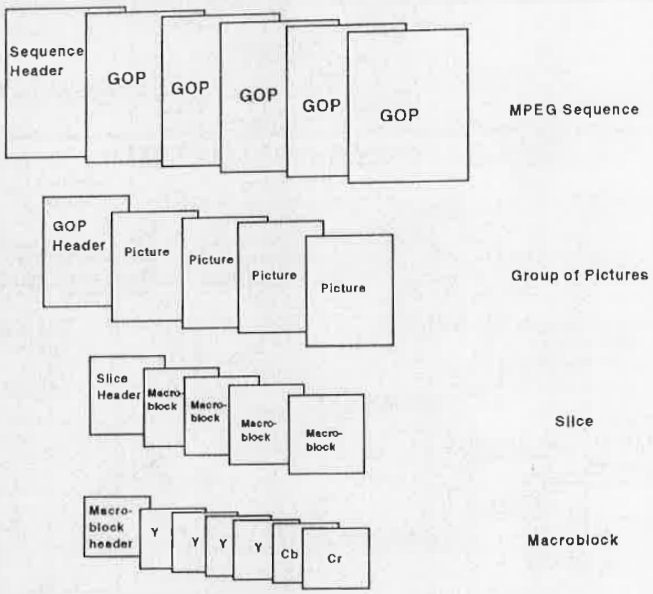


Figure 2.4 MPEG data stream.

I Picture Format

The I (Intraframe) picture format substantially corresponds to the JPEG format. These pictures are encoded by transformation into DCT space, quantization of the resultant coefficients, and entropy coding of the result. Transformation into DCT space is performed by an 8x8 DCT. Quantization is performed by reference to a user-loadable quantization table modified by a scale factor. This mechanism supports adaptive quantization at the cost of additional complexity - although 30% improvement in compression is claimed [PM93].

After quantization, the resulting coefficients are reordered in zig-zag order, run-length coded, variable-length coded, and entropy coded. The resulting data stream should show roughly JPEG levels of compression.

P Picture Format

The P (Predicted) picture format introduces the concept of motion compensation. Each macroblock is coded with a vector that predicts its value from an earlier I or P

frame. The decoding process copies the contents of the macroblock-sized data at the address referenced by the vector into the macroblock of the P frame currently being decoded. Five bits of resolution are reserved for the magnitude of the vector in each of the x and y directions, meaning that 1024 possible data blocks may be referenced by the predicted macroblock. However, eight possible magnitude ranges may be assigned to those five bits, meaning as many as 8192 macroblocks might have to be evaluated to exhaustively determine the best vector. Each evaluation might require testing as many as 384 pixels, and a further complexity is seen in performing fractional interpolation of pixels (vector motions as small as 1/2 pixel are supported). Finally, the difference between the prediction and the macroblock to be compressed may be encoded in like fashion to I frame encoding above.

B Picture Format

The B (Bidirectional prediction) picture format is calculated with two vectors. A backwards vector references a macroblock-sized region in the previous I or P frame, the forward vector references a macroblock-sized region in the next I or P frame. For this reason, I and P frames are placed in the coded stream before any B frames that reference them.

The macroblock-sized regions referenced by the motion compensation vectors are averaged to produce the motion estimate for the macroblock being decoded. As with P frames, the error between the prediction and the frame being encoded is compressed and placed in the bitstream. The error factor is decompressed and added to the prediction to form the B frame macroblock.

Many demanding technical issues are raised by the MPEG specification. These include fast algorithms for the DCT, fast algorithms for motion vector estimation, algorithms for adaptive quantization, and decompression in environments that allow some errors.

3

THE H.261/H.263 COMPRESSION STANDARD FOR VIDEO TELECOMMUNICATIONS

ITU has developed a video conferencing standard H.324 at very low bitrate for the General Switched Telephone Network (GSTN) and mobile radio [IT95a, IT95b, IT93]]. The H.324 is a recommendation for real-time voice, data, and video over V.34 modems on the GSTN telephone network. It consists of five documents: (1) H.324 systems, (2) H.223 multiplex, (3) H.245 control, (4) H.263 video codec, and (5) G.273 speech codec. The H.261 coding standard provides coded video at bit rates 64 Kbits/s and above, whereas the H.263 video coding standard, proposed for H.324, provides coded video around 16 Kbits/s.

Figure 3.1 shows a block diagram of a generic multimedia system, compliant to the H.324 standard. The system consists of terminal equipment, GSTN modem, GSTN network, multipoint control unit (MCU), and other system operation entities.

Video equipment includes cameras, monitors, and video processing units to improve compression. Audio equipment includes microphone, speakers, telephone instrument, and attached audio devices. Data application equipment includes computers, non-standardized data application protocols, telematic visual aids such as electronic whiteboards, etc.

GSTN network interface supports appropriate signaling, ringing functions and voltage levels in accordance with national standards.

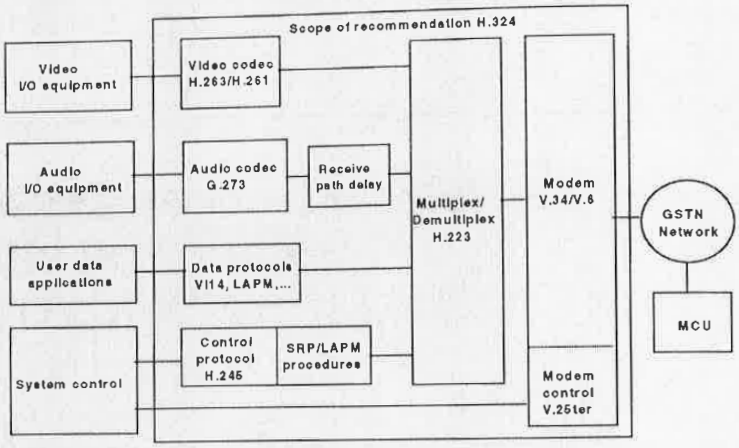


Figure 3.1. Block diagram of a generic H.324-compliant multimedia system.

3.1 PICTURE FORMATS FOR H.261/H.263 VIDEO CODECS

All H.324 terminals support both the H.263 and H.261 video codecs. For the H.261 algorithm two formats are defined: CIF and QCIF, while for the H.263 algorithm three additional formats are specified: SQCIF, 4CIF, and 16CIF.

The Common Intermediate Format (CIF) is a noninterlaced format, based on 352x288 pixels per frame at 30 frames per second. These values represent half the active lines of 625/25 television signal and the picture rate of a 525/30 NTSC signal. Therefore, 625/25 systems need only to perform a picture rate conversion, while NTSC systems need to perform only a line-number conversion.

Color pictures are coded using one luminance and two color-difference components (YCbCr format), specified by the CCIR 601 standard. The Cb and Cr components are subsampled by a factor of two on both horizontal and vertical directions, and have 176x144 pixels per frame. The picture aspect ratio for all five CIF-based

formats is 4:3. Table 3.1 summarizes the picture formats for H.261 and H.263 codecs.

Picture format	Luminance pixels	Maximum frame rate [f/s]	Video source rate	Average coded bit rate	H.261 codec	H.263 codec
SQCIF	128 x 96	30	1.3 Mb/s	26 Kb/s	Optional	Required
QCIF	176 x 144	30	9 Mb/s	64 Kb/s (px64 Kbps)	Required	Required
CIF	352 x 288	30	36 Mb/s	384 Kb/s (px64 Kbps)	Optional	Optional
4CIF	704 x 576	30	438 Mb/s	3-6 Mb/s	Not defined	Optional
16CIF	1408 x 1152	50	2.9 Gb/s	20-60 Mb/s	Not defined	Optional

Table 3.1. Picture formats for H.261 and H.263 video codecs.

3.2 H.261/H.263 VIDEO ENCODER

The H.261/H.263 video encoder combines intraframe and interframe coding to provide fast processing for on-the-fly video [Oku95, FSZ95, BK95, Fur95b]. The algorithm creates two types of frames:

(1) DCT-based intraframes, compressed using DCT, quantization, and entropy (variable-length) coding (similarly to JPEG) [Fur95a], and

(2) predictive interframes, compressed using Differential Pulse Code Modulation (DPCM) and motion estimation.

The block diagram of the video encoder is shown in Figure 3.2. The H.261/H.263 coding algorithm begins by coding an intraframe block and then sends it to the video multiplex coder. The same frame is then decompressed using the inverse quantizer and inverse DCT, and then stored in the frame memory for interframe coding.

During the interframe coding, the prediction based on the DPCM algorithm is used to compare every macro block of the actual frame with the available macro blocks of the previous frame, as illustrated in Figure 3.3.

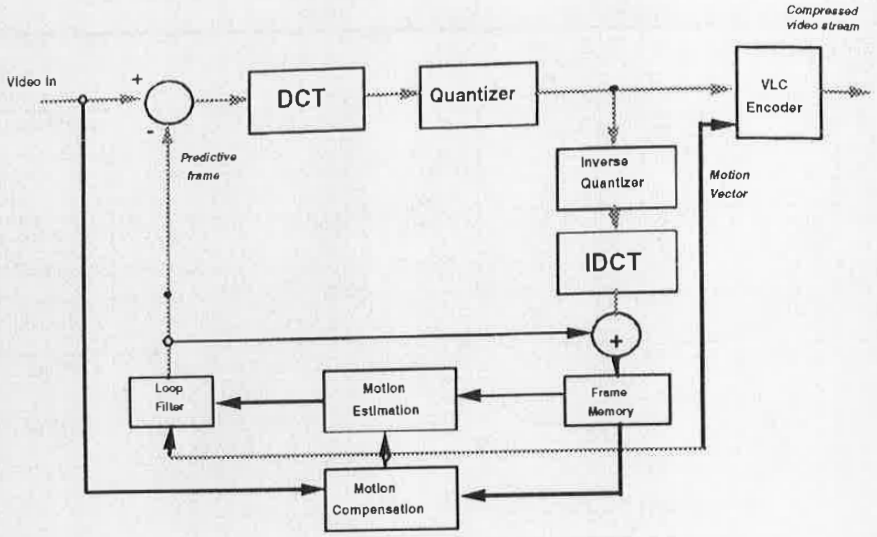


Figure 3.2. Block diagram of the H.261/H.263 video encoder.

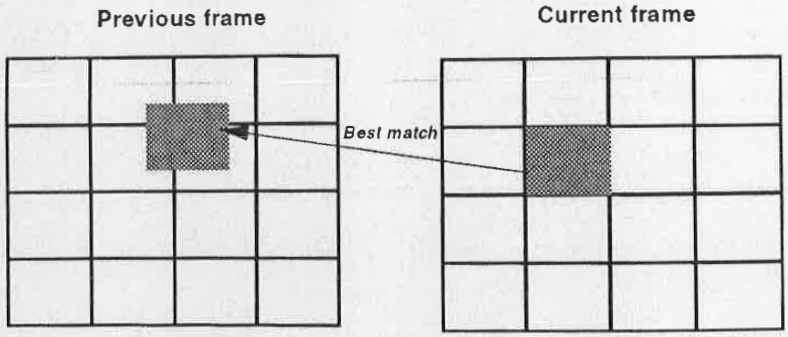


Figure 3.3 The principle of interframe coding in H.261/H.263 standard.
Each macro block (16 x 16 pixels) in the current frame is compared with macro blocks from the previous frame to find the best match.

To reduce the encoding delay, only the closest previous frame is used for prediction. Then, the difference, created as error terms, is DCT-coded and quantized, and sent to the video multiplex coder with or without the motion vector. At the final step, variable-length coding (VLC), such as Huffman encoder, is used to produce more compact code. An optional loop filter can be used to minimize the prediction error

by smoothing the pixels in the previous frame. At least one in every 132 frames should be intraframe coded, as shown in Figure 3.4.

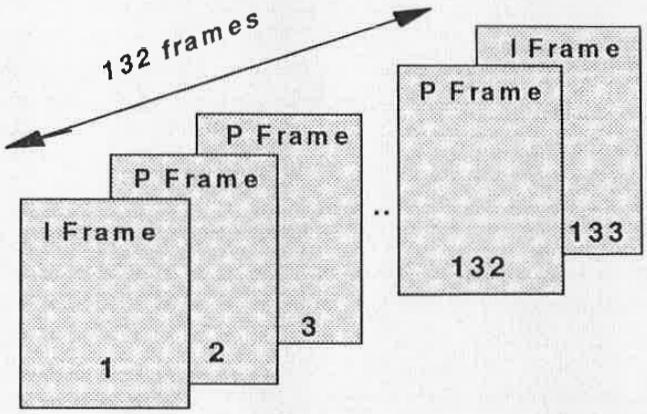


Figure 3.4 Types of frames in H.261/H.263 standard.
At least every 132-nd frame should be the I frame.

The compressed data stream is arranged in a hierarchical structure consisting of four layers: Pictures, Group of Pictures (GOPs), Macro Blocks (MB), and Blocks, as illustrated in Figure 3.5.

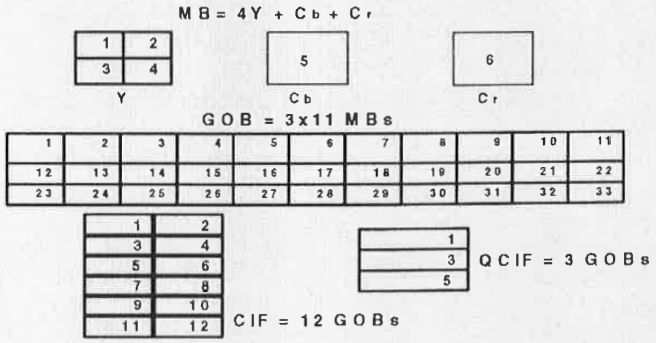


Figure 3.5 Hierarchical block structure of the H.261/H.263 video data stream.

3.3 THE H.261/H.263 VIDEO DECODER

The H.261/H.263 video decoder is shown in Figure 3.6. It consists of the receiver buffer, VLC decoder, inverse quantizer, inverse DCT, and the motion compensation, which includes frame memory and an optional loop filter [BSZ95, BK95, Fur95b].

In addition to the encoding and decoding of video, the audio data must also be compressed and decompressed. Special buffering and multiplexing/demultiplexing circuitry is required to handle the complexities of combining the video and audio.

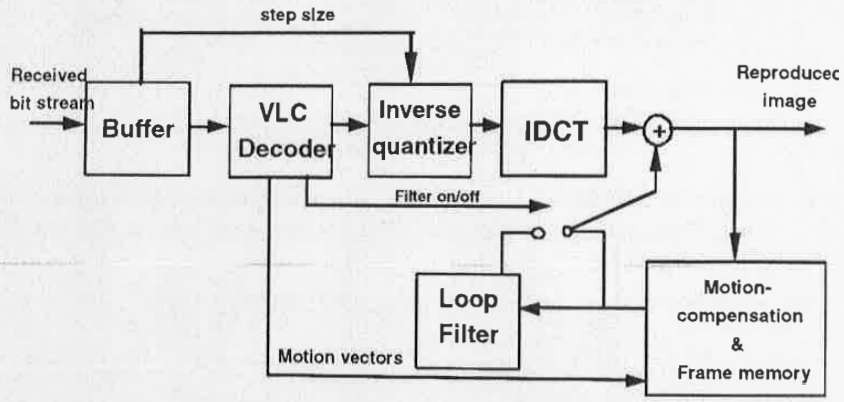


Figure 3.6 Block diagram of the H.261/H.263 video decoder.

7

ENTROPY CODING

Binary encoding of data is a natural means of representing computational data on modern digital computers. When the values to be encoded are uniformly distributed, this is an space-efficient means of representing the data as well. Information theory gives us several efficient methods of encoding "alphabets" where the likelihood of symbol occurrence varies symbol by symbol. Coding techniques that minimize space in the representation of random sequences of symbols (optimize space used in the representation of symbols based upon the probability the symbol) are known as entropy coding techniques.

There are two popular methods of entropy coding in the literature, Huffman coding and arithmetic coding. Huffman coding represents symbols with words of integer length while arithmetic coding is not limited to integer-length codes. Huffman coding is computationally less expensive to implement and typically gives compression ratios close to those of arithmetic coding. XYZ compression is developed to support Huffman coding [Huf52].

7.1 HUFFMAN CODING

Consider the problem of encoding the six symbols defined in Table 7.1. The amount of information transferred in a symbol A that occurs with probability p is:

$$I_A = \log_2\left(1/p_A\right) \tag{7.1}$$

where I_A is the number of bits required to express the amount of information conveyed by symbol A, and p_A is the probability that symbol A will occur.

The entropy of a code sequence is the average amount of information contained in each symbol of the sequence:

$$H = \sum_{s} p(s) * \log_2\left(1/p(s)\right) \tag{7.2}$$

where H is the entropy of the coding representation, and s ranges through all symbols in the alphabet of symbols.

Symbol	Probability	Information	Code
A	1/2	1 bit	0
B	1/4	2 bits	10
C	1/16	4 bits	1100
D	1/16	4 bits	1101
E	1/16	4 bits	1110
F	1/16	4 bits	1111

Table 7.1 Symbols and their associated Huffman code.

The entropy of the sequence represents the lower bound of the space needed to communicate the information contained in the sequence. A fixed word length of three bits may be used to represent the six symbols in Table 7.1. Using the Huffman coding representation, we get an average code length of 2, which for these probabilities happens also to be the entropy, or lower limit of the average code length:

$$H = \left(1/2\right)\times 1 + \left(1/4\right)\times 2 + \left(1/16\right)\times 4 + \left(1/16\right)\times 4 + \left(1/16\right)\times 4 + \left(1/16\right)\times 4$$
$$= 2 \tag{7.3}$$

Assignment of Huffman codes is done by developing a Huffman coding tree, as illustrated in Figure 7.1. The tree is developed "left to right" (or bottom to top). Symbols are listed in decreasing order of probability. Iteratively, the two adjacent branches whose sum is least are combined into a new branch, until the tree is completed. The tree is then traversed and labeled.

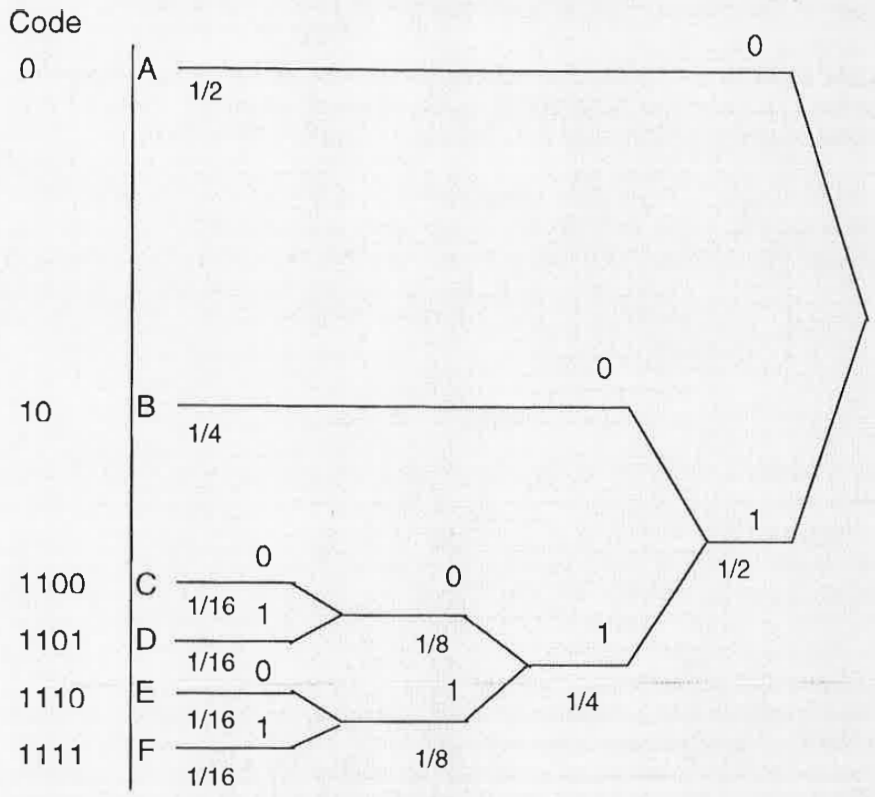


Figure 7.1 Huffmand coding tree - an example.

This algorithm computes Huffman codes for arbitrary sequences of symbols, and will work regardless of their relative probabilities (real cases don't show the simple logarithmic behavior of this example). The algorithm is simple enough (O(NlogN)) to form the kernel of a real-time algorithm for the adaptive entropy coding of images. Huffman coding is used in the JPEG and MPEG specification

7.2 USE OF ENTROPY CODING IN JPEG AND MPEG

Once Huffman coding has been chosen to represent the data, the compression attained by the system is influenced by the choice of alphabet. Several different alphabets have been defined in the JPEG, MPEG-1, and MPEG-2 standards.

Variable-length coding assigns a length prefix to each word, and follows it with a value code, as shown in Table 7.2. If the values to be represented are small on average, this technique helps reduce the number of bits required to represent them.

Number of bits	Range of encoded values
1	-1,1
2	-3..-2, 2..3
3	-7,..-4, 4..7
4	-15..8, 8..15
5	-31,16, 16..31
6	-63..32, 32..63
7	-127..64, 64..127
8	-255..-128, 128..255
9	-511..-256, 256..511
10	-1023..-512, 512..1023
11	-2047..-1024, 1024..2048
12	-4095..-2047, 2047..4095
13	-8191..-4096, 4096..8191
14	-16383..-8192, 8192..16383
15	-32767..-16384, 16384..32767
16	-65535..-32768, 32768..65535

Table 7.2 VLC length code and range of encoded values.

The concept of run-length coding is also introduced. In its more general form, run-length coding encodes a "run" of consecutive equal values into a length code followed by the value of the run. In the case of JPEG, only zeroes are run-length coded. The run-length code stands alone, and the zero value is implied.

JPEG introduced a modified Huffman coding scheme that introduces many of the concepts later used by the MPEG specifications. JPEG constructs symbols from sequences of DCT components. Two fundamental concepts are used: variable length coding, and run-length coding. JPEG words are formatted into a run length, a VLC length, and a VLC value. The run length and VLC length are combined to form the

JPEG alphabet, given in Table 7.3. Four bits are allocated to each length code, giving 256 symbols in the alphabet.

Run \ VLC	0	1	2	3	4	5	6	7
0	EOB	01	02	03	04	05	06	07
1	N/A	11	12	13	14	15	16	17
2	N/A	21	22	23	24	25	26	27
3	N/A	31	32	33	34	35	36	37
4	N/A	41	42	43	44	45	46	47
5	N/A	51	52	53	54	55	56	57
6	N/A	61	62	63	64	65	66	67
7	ZRL	71	72	73	74	75	76	77

Table 7.3 The JPEG alphabet.

The JPEG run length and VLC words are combined and treated as symbols to be Huffman encoded. The value 0 is never coded, but is treated as part of a run. Two special symbols are introduced, EOB and ZRL. EOB is the End Of Block symbol, and marks the end of each block, implying a run of zeroes to fill the rest of the block. The ZRL code is the Zero Run Length code, and represents a run of exactly 16 zeroes. This code is used to segment runs of over 16 zeroes.

DCT coefficients are arranged in a simple "zig-zag" order (Table 7.4) intended to concentrate the zero components and increase the run lengths. The variance of each DCT component predicts the probability that component will be 0 (smaller variances predict 0). Higher frequency components tend to have larger variances, and so JPEG orders the components into increasing order of frequency before encoding.

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

Table 7.4 The JPEG zig-zag order of DCT coefficients.

MPEG-1 encodes a different alphabet - the run length, VLC length, and the VLC value are combined into a single symbol of the alphabet. Even worse, MPEG-1 allows run lengths of up to 63 zeroes (the maximum possible as there are only 64 DCT coefficients), resulting in a proliferation of symbols. However, DCT components are limited in value to the range [-256..255]. The number of symbols in the alphabet now becomes the number of possible zero runs multiplied by the number of VLC values plus the number of special symbols (EOB). This makes the total number of symbols 32K+1. MPEG-1 zero run length codes are shown in Table 7.5.

Run Length	Code
0	000000
1	000001
2	000010
.	.
62	111110
63	111111

Table 7.5 MPEG-1 zero run length codes.

In order to limit the length of code words to 16 bits, MPEG-1 introduces a new symbol, escape (ESC). The ESC symbol triggers the decoder to look for a run-length code followed by a level code. Upon examination of sample video, the most common 222 VLC codes were selected as the alphabet for MPEG-1. All other VLC codes are coded with the escape mechanism. MPEG-1 level codes are shown in Table 7.6.

The MPEG-2 encoding scheme is virtually identical to the MPEG-1 encoding scheme, except that the escape levels are fixed at 12 bits, while those of MPEG are either 8 or 16 bits.

7.3 ADAPTIVE HUFFMAN CODING

This section describes a modification of the JPEG encoding scheme well suited to encoding motion video. The JPEG alphabet is extended with the ESC symbol. The alphabet to be encoded will be composed of all 256 combinations of 4 bit run length and 4 bit VLC codes, ZRL, and EOB.

LEVEL	CODE
-255	1000 0000 0000 0001
-254	1000 0000 0000 0010
.	.
-129	1000 0000 0111 1111
-128	1000 0000 1000 0000
-127	1000 0001
-126	1000 0010
.	.
-2	1111 1110
-1	1111 1111
1	0000 0001
2	0000 0010
.	.
126	0111 1110
127	0111 1111
128	0000 0000 1000 0000
129	0000 0000 1000 1001
.	.
254	0000 0000 1111 1110
255	0000 0000 1111 1111

Table 7.6 MPEG-1 level codes.

Statistics for the group of frames to be decoded are gathered. The count of the occurrence of each symbol in the alphabet within the group of frames is collected, and a table of the counts is collected. The count for ESC is set to 0.

The standard Huffman coding procedure, described in Section 7.1, is applied to the data. Two additional tables are constructed, one for the temporary storage of symbols waiting to be coded (representing the root of each branch of the Huffman coding tree), and one for the storage of the symbols that are in the process of being coded (the branches of the tree), as illustrated in Figure 7.2.

Entries in the Encoder Alphabet Table (Table 7.7) whose count is non-zero are copied into the Root Table, setting the #bits field to 0, and the left and right subtrees to none. The procedure to construct the Huffman code tree is then followed.

SYMBOL	COUNT	#BITS	LEFT	RIGHT

Root Table

SYMBOL	COUNT	#BITS	LEFT	RIGHT

Branch Table

Figure 7.2 Format of Root Table and Branch Table

The Root Table is sorted by the count field. The two symbols (branches) with smallest count are removed from the Root Table and added into Branch Table. A new branch entry is created to be added into the Root Table. The left and right fields of the new entry will point to the two branches just added into the Branch Table. The count field of the new entry will be set to the sum of the count fields of the two branches just added to the Branch Table, and the new branch is added to the Root Table in its sorted position. This process is repeated until there is exactly one entry in the Root Table.

The Huffman coding tree is traversed, and the number of bits to be assigned to each code is accumulated. Any symbol that requires more than 12 bits to encode is removed from the Encoder Alphabet Table (by setting its count field to 0), and the value of its count field is added to the count field of the ESC symbol.

A new Huffman Coding tree is built - a new Root Table is built from the Encoder Alphabet Table, a new Huffman Coding Table is built. When this tree is traversed, all entries with less than 12 bits have their codes written to the Encoder Alphabet Table. The Encoder Alphabet Table is now ready for use to encode the frame data.

Encoding is straightforward for entries with less than 12 bits. Entries with over 12 bits are coding using an escape coding procedure - the ESC symbol is written, then the 8 bit run length / variable length field, then the VLC value to be encoded.

SYMBOL (run/vlc)	COUNT	CODE
0/1	nnn	
0/2	nnn	
.	.	
0/F	nnn	
1/1	nnn	
1/2	nnn	
.	.	
1/F	nnn	
F/1	nnn	
F/2	nnn	
.	.	
F/F		
ZRL	nnn	
EOB	nnn	
ESC	0	

Table 7.7 Encoder Alphabet Table